MTP (Music Tour Success Prediction) Progress Report Team #125: JJ. BASK CSE6242 Fall 2023

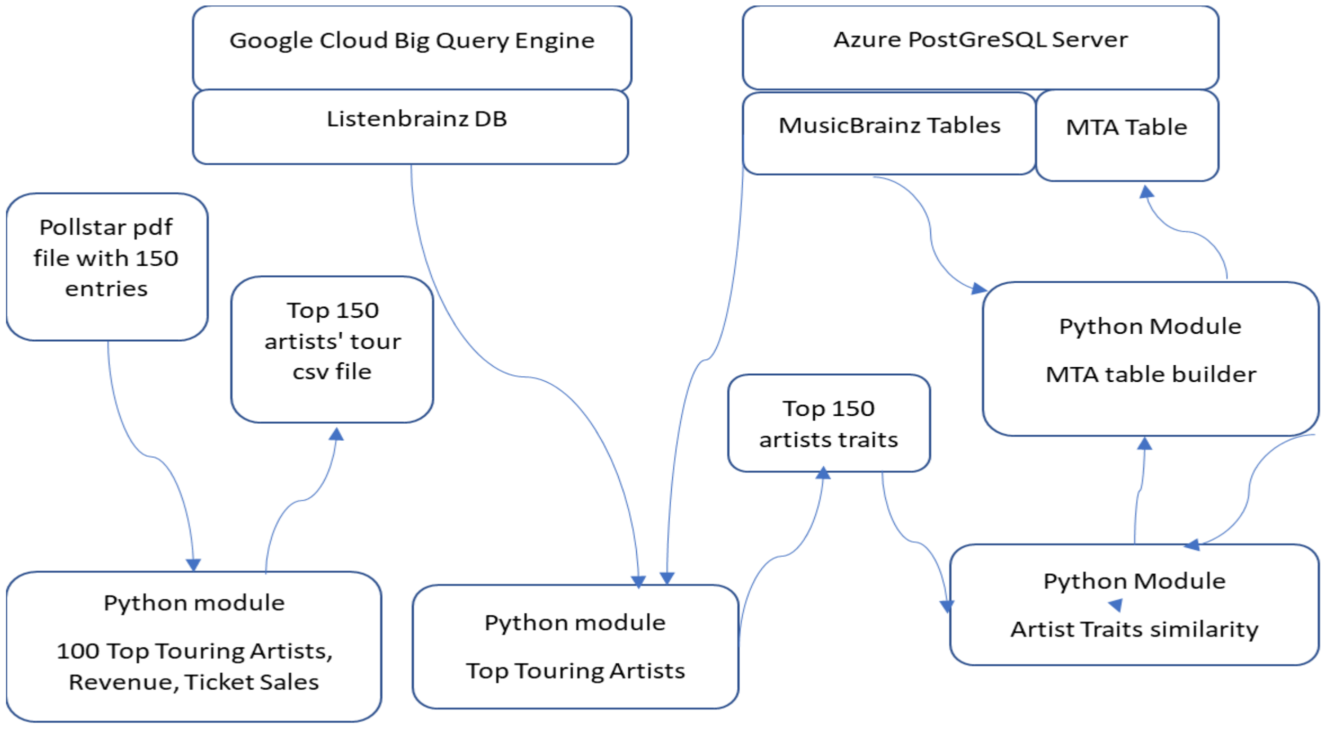
Team #125: JJ. BASK. Team Members

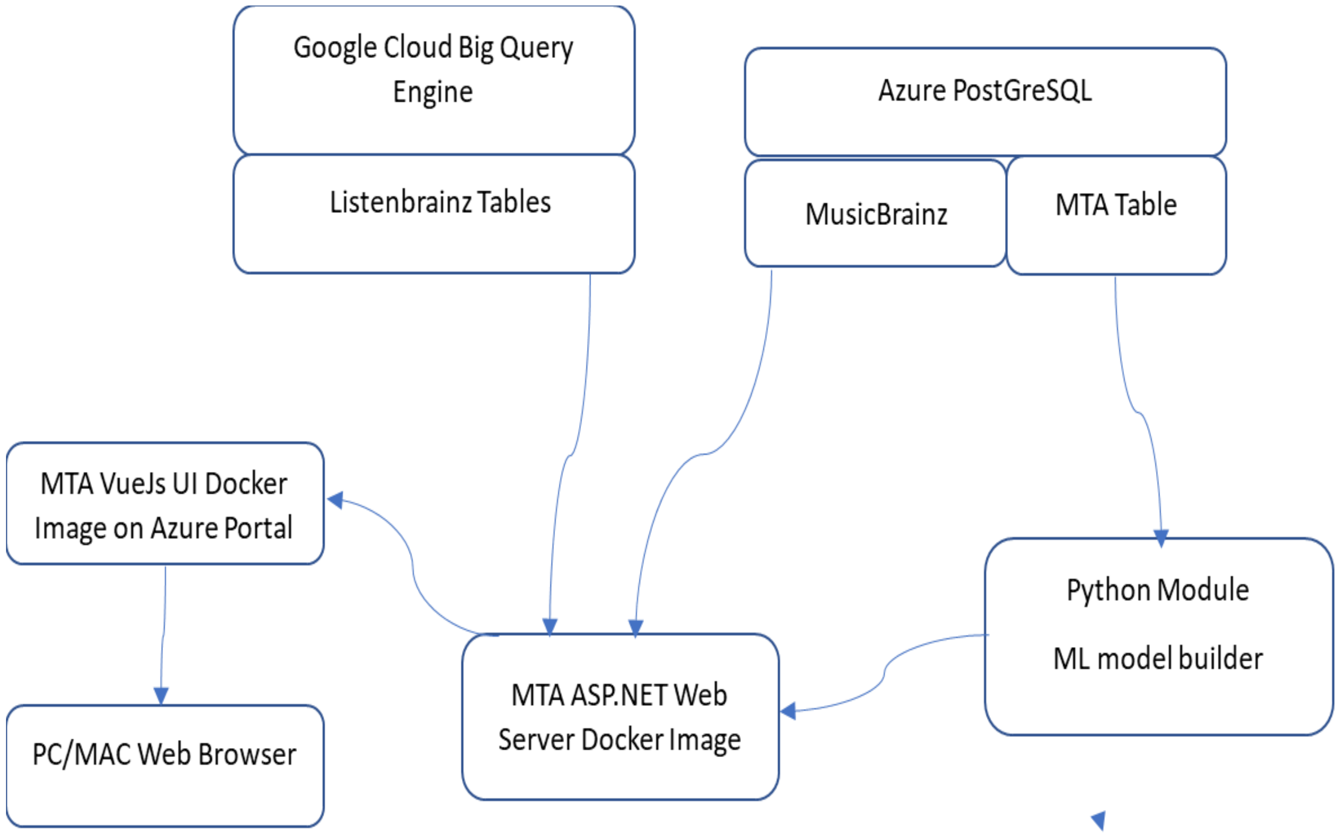
Sai Krishna Pothana (spothana3), Jordan Gewing-Mullins (jag31), Bikash Kumar Jha (bjha32), Alexander Muriithi (amuriithi3), Katherine Bae (kbae41)

Project

Our objective is to visualize the music industry given data about artists, releases, labels, and the relationships between them. We also aim to predict the success of a music tour given data about artists, their work and the relationships between them, ticket sales and music consumption pattern. The end goal of the project is to create a cloud-based application which would assist music aficionados and music industry's budding entrepreneurs in visualizing the trends in the music industry for choosing the right kind of music for a particular region in the world. It would also help music tour investors in choosing the right artist for their investment. The revenue for the product would be from advertising, music artist registration and their nascent music productions. Typically, investors in like US do not know anything about artists in other countries like India, China, Korea etc. … so they can use our application to invest in the music tours of artists from other countries and have a probability of good returns. The algorithm presented here is patentable, and we intend to file a patent application.

Design


Google Cloud Big Query Engine
Listenbrainz DB
Azure PostGreSQL Server
MusicBrainz Tables
MTA Table
Pollstar pdf file with 150 entries
Top 150 artists' tour csv file
Python Module
MTA table builder
Top 150 artists traits
Python module
100 Top Touring Artists, Revenue, Ticket Sales
Python module
Top Touring Artists
Python Module
Artist Traits similarity


Google Cloud Big Query Engine
Listenbrainz Tables
Azure PostGreSQL
MusicBrainz
MTA Table
MTA VueJs UI Docker Image on Azure Portal
PC/MAC Web Browser
MTA ASP.NET Web Server Docker Image
Python Module
ML model builder

Data Collection

MusicBrainz

MusicBrainz data set is a music encyclopedia that provides information on artists, albums, tracks, genres, and much more. MusicBrainz captures information about artists, their recorded works, and the relationships between them. Recorded works entries capture at a minimum the album title, track titles, and the length of each track.

As the data set is huge and not hosted by any of the popular cloud platforms, we created a PostgreSQL server with all the tables from musicbrainz

Steps for creating a Musicbrainz PostgreSQL server on Azure

- Created a PostgreSQL server with an empty database collection named musicbrainz and configure it to be accessible for public with SQL authentication
  - **PGHOST=musicbrainz.postgres.database.azure.com**
  - **PGUSER=musicbrainz**
  - **PGPORT=5432**
  - **PGDATABASE=musicbrainz**
  - **PGPASSWORD="{your-password}"**
- Downloaded PostgreSQL dumps for various tables in musicbrainz database from **https://ftp.osuosl.org/pub/musicbrainz/data/fullexport/20231018-002451/**
- Downloaded **https://github.com/lalinsky/mbslave** python module and modified it to tune the environment variables for our azure PostgreSQL server.
- Using mbslave python module python module, created musicbrainz tables as per musicbrainz schema
- Using mbslave python module, data dumps were uploaded for individual tables to musicbrainz database hosted at **musicbrainz.postgres.database.azure.com.**
- Tested the connection using Azure Data studio application local PCs

ListenBrainz

The ListenBrainz data set is a time series music metadata for users who report the music to which they have recently listened. Artist, album, recording, listened at timestamp and ancillary data that is submitted to the ListenBrainz server is streamed live to Big Query and should appear within a few seconds of submission to ListenBrainz. The data we gather and publish via Big Query is defined in our API documentation. This data is a valuable resource for discovering users' music listening habits and should be a valuable source for creating music recommendation systems. This dataset can be accessed via python, Azure data studio and a .NET library

- **Dataset source: https://ftp.osuosl.org/pub/musicbrainz/data/json-dumps**
- **Cloud service: Google Cloud Big Query**

Artist tour revenue and ticket sales

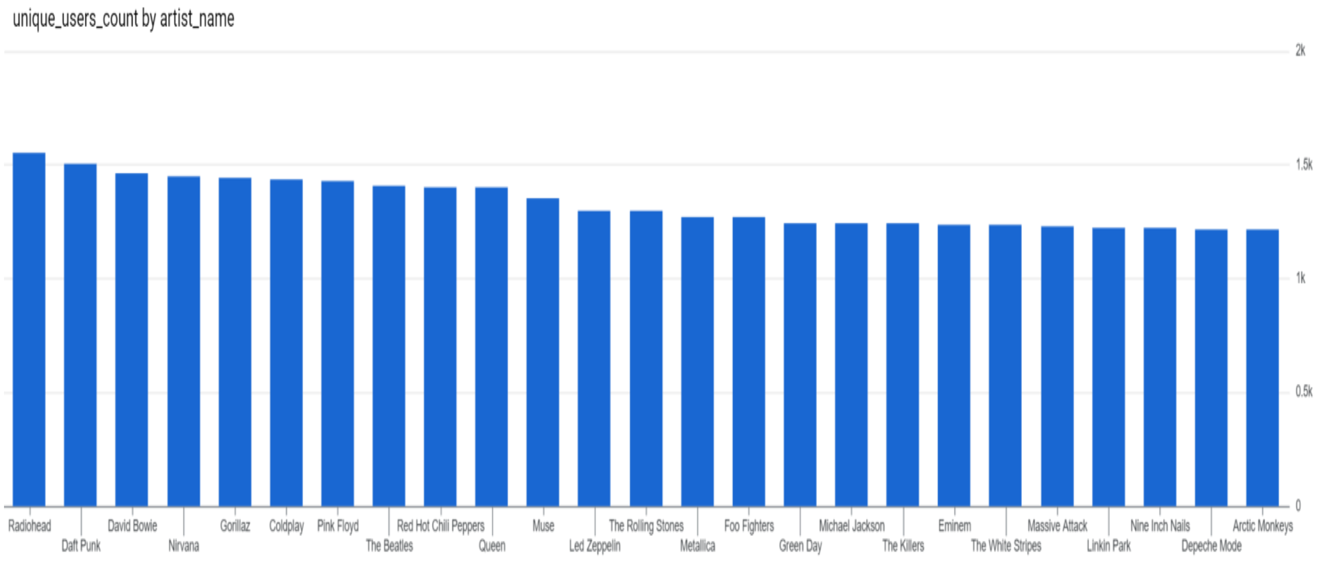
The ticket and sales info from poll star is expensive for a student project so we decided to predict the predict revenue and sales of artist music tour from a small set of entries provided by poll star free of cost. From Pollstar, download top touring artists of the Pollstar Era and create a table named TOP_TOURING_ARTIST_REVENUE_SALES (150 entries)

Data Visualization

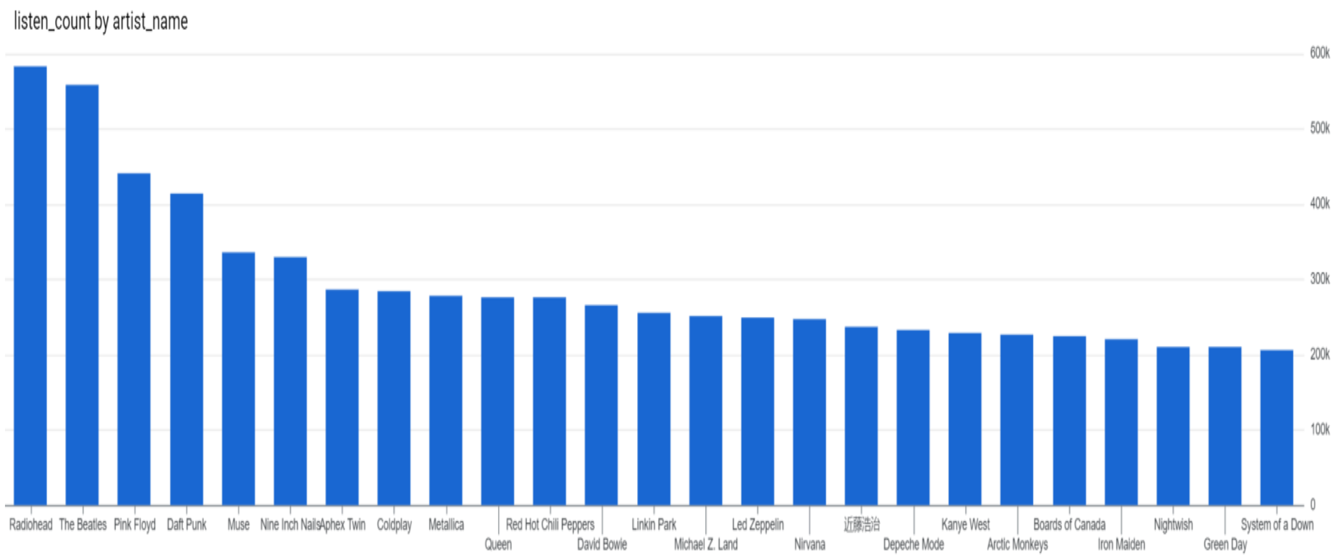
Musicbrainz visualizations are created using Azure Data studio by querying musicbrainz.postgres.database.azure.com and using the Azure Data studio's charting extension software

Listenbrainz visualizations are created using Google clouds Big Query engine.

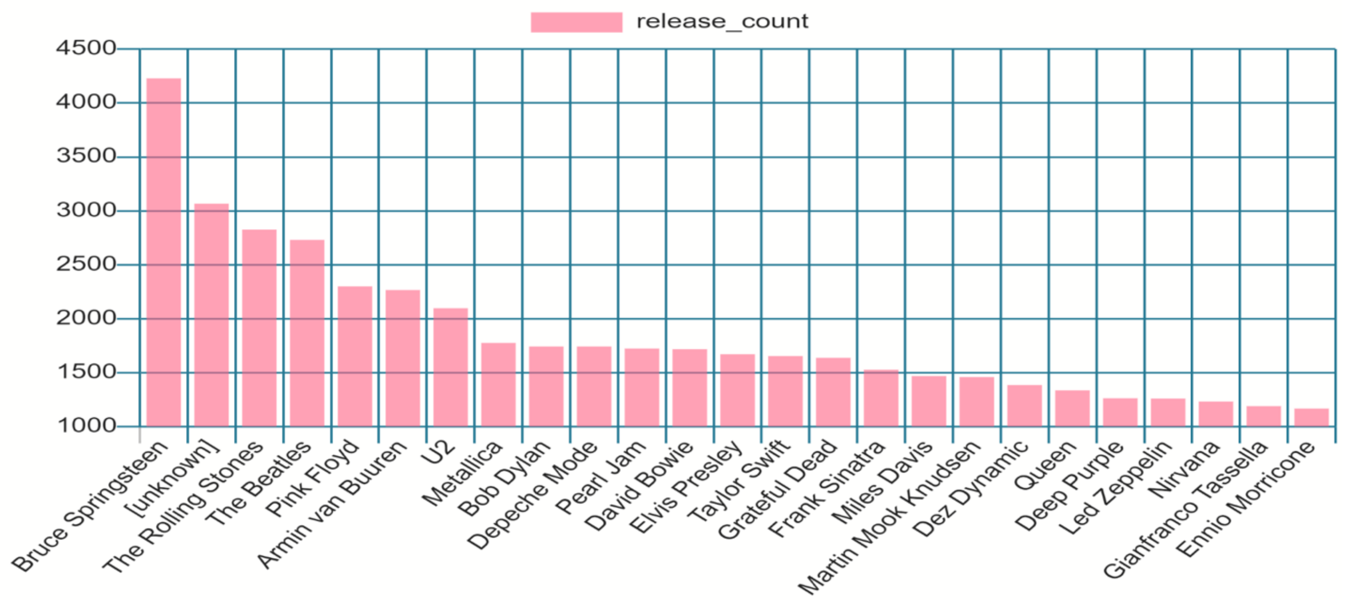
Top 25 Artist vs Listeners Count



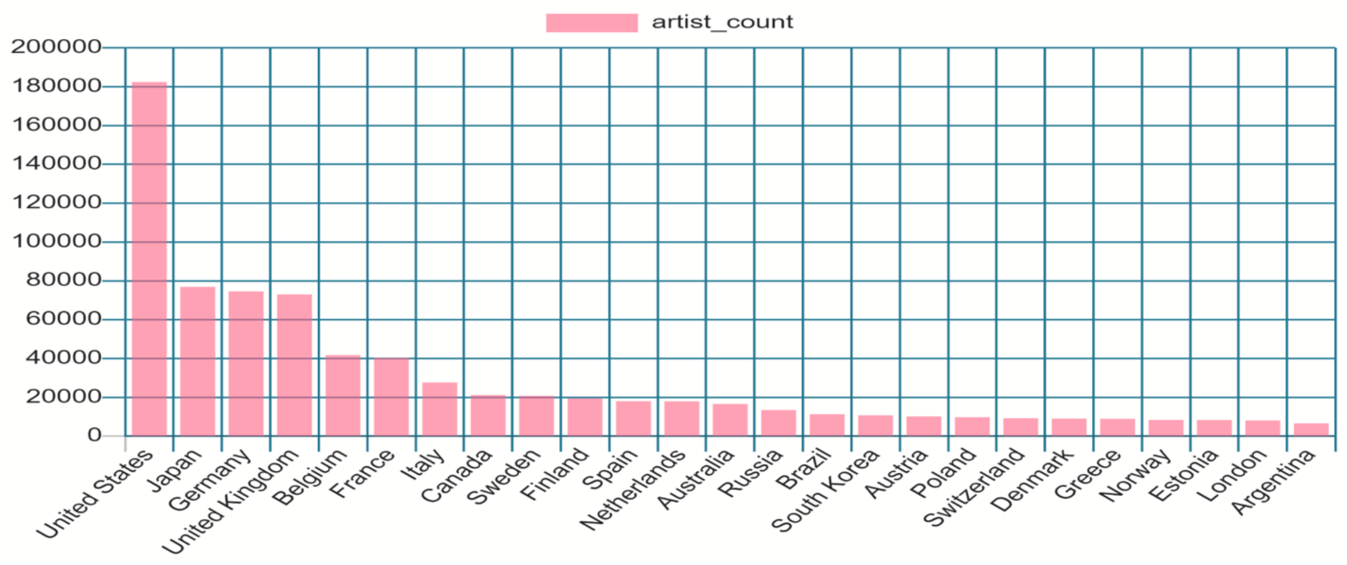
Top25 Artists Vs Listen Count



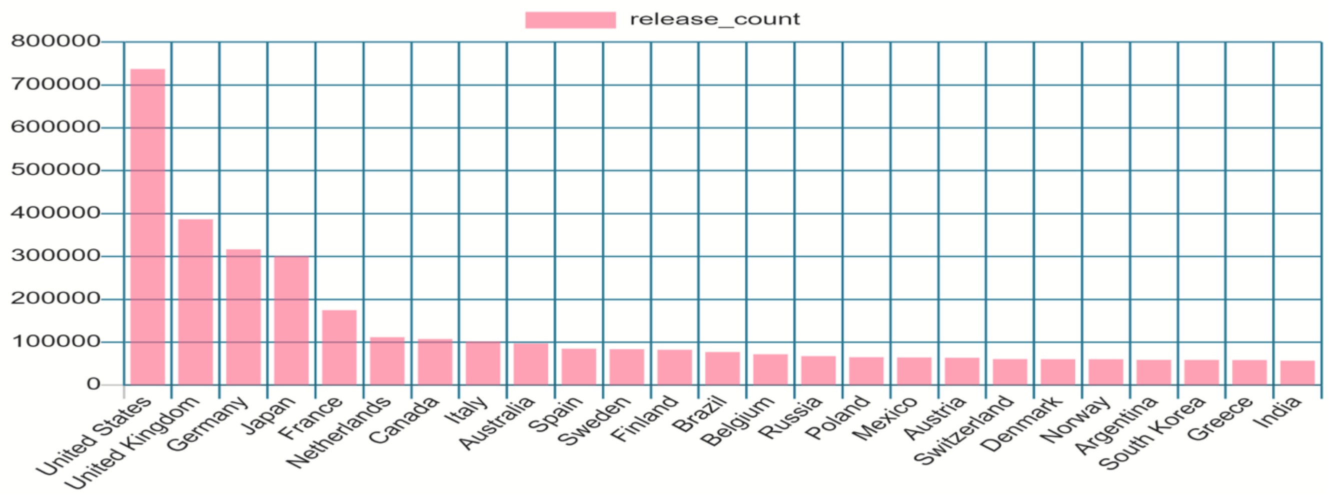
Top 25 Artists vs their music release count



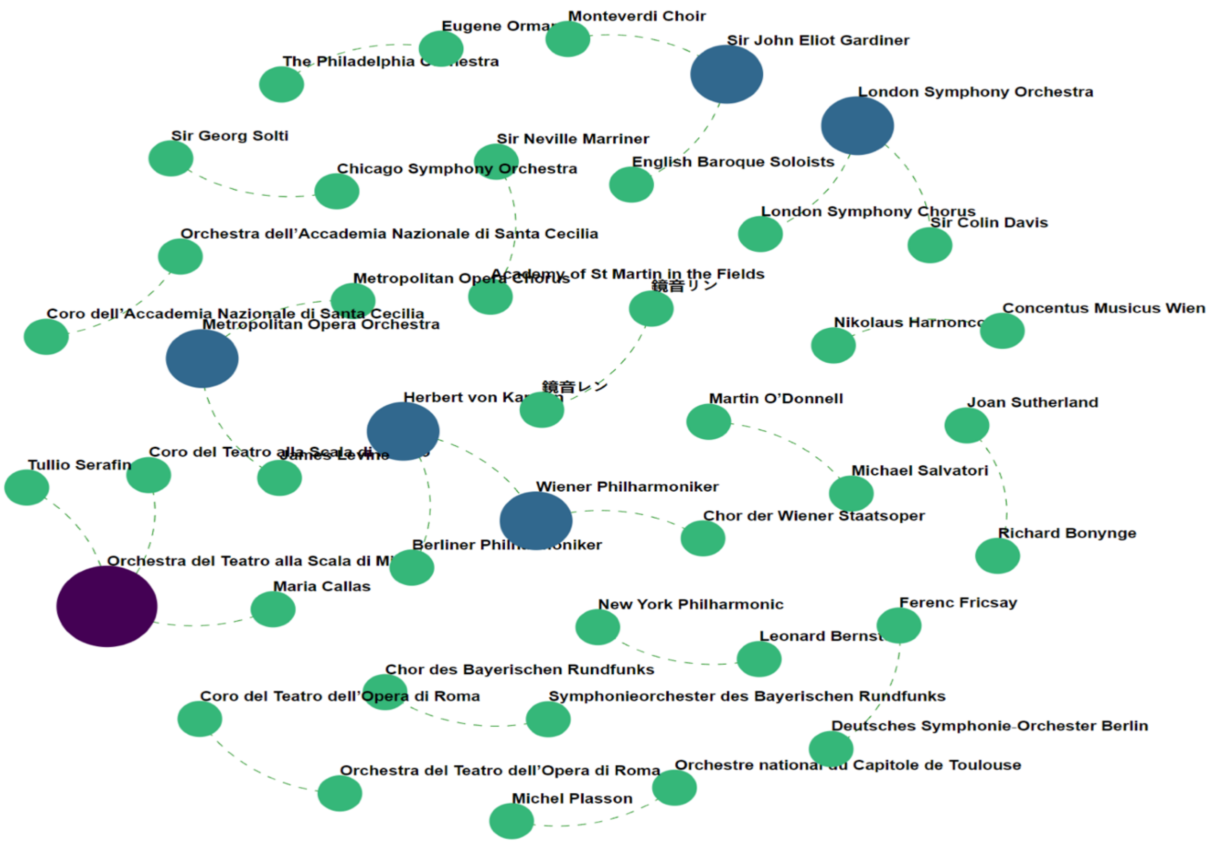
Top 25 countries with largest number of artists



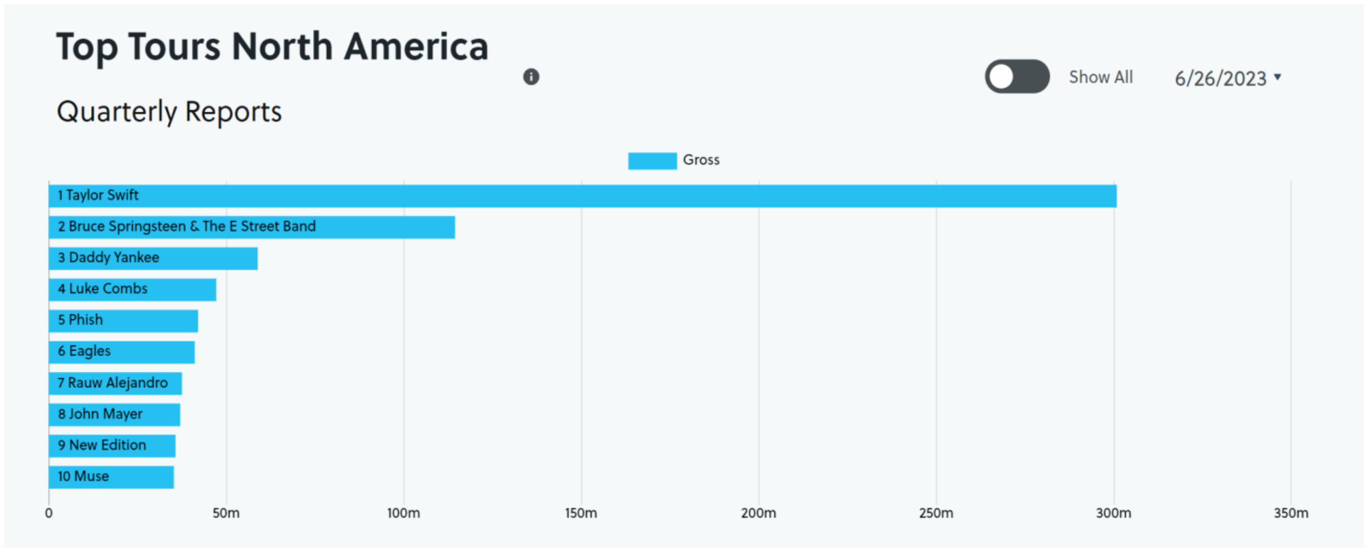
Top 25 countries with largest number of music releases



Top 25 artist who collaborated the most with other artists


Eugene Orma
Monteverdi Choir
The Philadelphia Orchestra
Sir John Eliot Gardiner
London Symphony Orchestra
Sir Georg Solti
Sir Neville Marriner
Chicago Symphony Orchestra
English Baroque Soloists
London Symphony Chorus
Sir Colin Davis
Orchestra dell'Accademia Nazionale di Santa Cecilia
Metropolitan Opera Chorus
Academy of St Martin in the Fields
鏡音リン
Coro dell'Accademia Nazionale di Santa Cecilia
Metropolitan Opera Orchestra
Concentus Musicus Wien
Nikolaus Harnonco
鏡音レン
Herbert von Karajan
Martin O'Donnell
Joan Sutherland
Tullio Serafin
Coro del Teatro alla Scala di Milano
James Levine
Michael Salvatori
Wiener Philharmoniker
Chor der Wiener Staatsoper
Richard Bonynge
Orchestra del Teatro alla Scala di Milano
Berliner Philharmoniker
Maria Callas
New York Philharmonic
Ferenc Fricsay
Leonard Bernstein
Chor des Bayerischen Rundfunks
Coro del Teatro dell'Opera di Roma
Symphonieorchester des Bayerischen Rundfunks
Deutsches Symphonie-Orchester Berlin
Orchestra del Teatro dell'Opera di Roma
Orchestre national du Capitole de Toulouse
Michel Plasson

Top 10 music tours by revenue from Pollstar



Data Collection and Pre-processing

Revenue and sales prediction using a similarity algorithm.

The ticket and sales info from poll star is expensive for a student project so we decided to predict the predict revenue and ticket sales for an artist music tour from a small set of entries provided by poll star free of cost. **TOP_TOURING_ARTIST_REVENUE_SALES table** provides just artist name, revenue and tickets sold. Using Musicbrainz and ListenBrainz, we will take the top tour earning artists and add Genre, Collaboration count, Artist + collaborating artists' Instruments, Release count, Listen User counts, active years, listen count of all their releases to another database table named **TOP_TOURING_ARTISTS_TRAITS**

A similarity match would be made between **INPUT_ARTIST** (Genre, Collaboration count, Artist's + collaborating artists' Instruments, Release count, Listen User counts, active years, listen count) whose tour revenues are not known and top touring artists. Whichever entry in **TOP_TOURING_ARTISTS table** has a maximum match with the provided **INPUT_ARTIST**, that artist's revenue and sales from the TOP**_TOURING_ARTIST_REVENUE_SALES as** a percentage of match would be assigned to **INPUT_ARTIST's** revenue and sales This process will be repeated for all the artists from Musicbrainz.

Similarity algorithm

Here is a basic outline on how we would do it

Preprocessing:

Normalize the data if they are numerical to ensure they are on the same scale. Convert categorical data to numerical values if possible, using techniques like one-hot encoding or label encoding. For textual data, convert them into numerical vectors using techniques like TF-IDF or word embeddings.

Compute Similarity:

For numerical data, compute the Euclidean distance or cosine similarity. For categorical data, compute similarity based on matching categories. For textual data, compute the cosine similarity of the numerical vectors.

Aggregate Similarity Scores:

If each row has multiple columns of diverse types (e.g., numerical, categorical, textual), compute the similarity score for each column separately and then aggregate them to get a final score for the entire row.

Percentage Match:

Once the similarity scores are available, convert them to a percentage match, normalize the scores such that they lie between 0 to 100.

MTA Table

Independent features

Genre, Collaboration count, Artist + collaborating artists' Instruments, Release count, Listen User counts, Active years, Listen count of all their releases.

Dependent feature

Predicted Sales and revenue for each artist would be converted into categorical values like high success rate, medium success rate, and low success rate.

Feature Extraction

Independent variables

Genre, Collaboration count, Artist + collaborating artists' Instruments, Release count, Listen User counts, Active years, Listen count of all their releases

Dependent variable – Probability of success

High, Medium, Low

ML Model building and Evaluation

Model building would be done using python scikit-learn module or python spark ml module or azure ml.

SaaS application

Frontend

VueJS framework would be used to create a frontend application that would provide

- UI for visualizing different charts dynamically using user provided input
- UI for taking user input to be used in ML model inferences and for displaying the results from the ML model

Backend

An ASP.NET based back-end web application would be used to provide

- REST API for VueJS frontend
- Interface to the database tables from Azure and GCP

Hosting

The plan is to host databases, frontend, and backend on Azure portal